

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Atsushi MASUDA

GAU:

SERIAL NO: New Application

EXAMINER:

FILED: Herewith

FOR: DESIGN METHOD OF LOGIC CIRCUIT

REQUEST FOR PRIORITY

COMMISSIONER FOR PATENTS
ALEXANDRIA, VIRGINIA 22313

SIR:

- ☐ Full benefit of the filing date of U.S. Application Serial Number _____, filed _____, is claimed pursuant to the provisions of 35 U.S.C. §120.
- ☐ Full benefit of the filing date(s) of U.S. Provisional Application(s) is claimed pursuant to the provisions of 35 U.S.C. §119(e): Application No. _____ Date Filed _____
- ☒ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of 35 U.S.C. §119, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

<u>COUNTRY</u>	<u>APPLICATION NUMBER</u>	<u>MONTH/DAY/YEAR</u>
Japan	2002-354208	December 5, 2002

Certified copies of the corresponding Convention Application(s)

- ☒ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee
- ☐ were filed in prior application Serial No. _____ filed _____
- ☐ were submitted to the International Bureau in PCT Application Number _____
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.
- ☐ (A) Application Serial No.(s) were filed in prior application Serial No. _____ filed _____; and
- ☐ (B) Application Serial No.(s) _____
☐ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.



Marvin J. Spivak

Registration No. 24,913

C. Irvin McClelland
Registration Number 21,124



22850

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application:

2002年12月 5日

出願番号
Application Number:

特願2002-354208

[ST.10/C]:

[JP2002-354208]

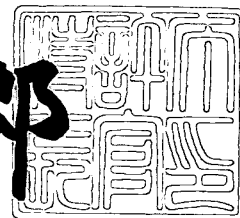
出願人
Applicant(s):

株式会社東芝

2002年12月27日

特許庁長官
Commissioner,
Japan Patent Office

太田信一郎



出証番号 出証特2002-3102614



【書類名】 特許願

【整理番号】 ASB023110

【提出日】 平成14年12月 5日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 17/50

【発明の名称】 論理回路設計方法、論理回路設計プログラムおよび論理回路設計装置

【請求項の数】 15

【発明者】

【住所又は居所】 神奈川県川崎市幸区小向東芝町1番地 株式会社東芝マイクロエレクトロニクスセンター内

【氏名】 増田 篤司

【特許出願人】

【識別番号】 000003078

【氏名又は名称】 株式会社 東芝

【代理人】

【識別番号】 100083806

【弁理士】

【氏名又は名称】 三好 秀和

【電話番号】 03-3504-3075

【選任した代理人】

【識別番号】 100068342

【弁理士】

【氏名又は名称】 三好 保男

【選任した代理人】

【識別番号】 100100712

【弁理士】

【氏名又は名称】 岩▲崎▼ 幸邦



【選任した代理人】

【識別番号】 100100929

【弁理士】

【氏名又は名称】 川又 澄雄

【選任した代理人】

【識別番号】 100108707

【弁理士】

【氏名又は名称】 中村 友之

【選任した代理人】

【識別番号】 100095500

【弁理士】

【氏名又は名称】 伊藤 正和

【選任した代理人】

【識別番号】 100101247

【弁理士】

【氏名又は名称】 高橋 俊一

【選任した代理人】

【識別番号】 100098327

【弁理士】

【氏名又は名称】 高松 俊雄

【手数料の表示】

【予納台帳番号】 001982

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 論理回路設計方法、論理回路設計プログラムおよび論理回路設計装置

【特許請求の範囲】

【請求項 1】 論理回路のアルゴリズムを、複数の演算子を有する動作記述から、前記演算子を実行する演算ノードを実行順に配列したデータフロー図に変換するステップと、

前記データフロー図に実行ステップを割り当て、前記演算ノードからの出力データを記憶するレジスタを前記実行ステップの実行後に挿入するステップと、

前記演算ノードとして機能する演算器と前記レジスタとして機能する記憶素子を有する前記論理回路のデータパスと、前記データパスの制御情報を生成するステップと、

前記演算子を実行する前記演算器を前記演算子から検索可能であり、前記記憶素子の機能をする前記レジスタが記憶するデータを出力する前記演算子を前記実行ステップと前記記憶素子から検索可能である演算子演算器データベースを生成するステップとを有することを特徴とする論理回路設計方法。

【請求項 2】 前記動作記述と前記データフロー図に基づいて、前記演算ノードで実行する前記演算子を前記演算ノードから検索可能であり、前記レジスタが記憶するデータを出力する前記演算ノードで実行する前記演算子を前記レジスタから検索可能であるノード演算子データベースを生成するステップと、

前記データフロー図と前記データパスに基づいて、前記演算ノードとして機能する前記演算器を前記演算ノードから検索可能であり、前記記憶素子の機能をする前記レジスタを前記実行ステップと前記記憶素子から検索可能であるノード演算器データベースを生成するステップをさらに有し、

前記演算子演算器データベースを生成するステップが、前記ノード演算子データベースと前記ノード演算器データベースに基づくこととを特徴とする請求項 1 に記載の論理回路設計方法。

【請求項 3】 前記データパスと前記制御情報に基づいて、前記論理回路のレジスタ転送レベル記述を生成するステップと、

前記動作記述に入力データを代入し、前記演算子からの出力データを計算するステップと、

前記レジスタ転送レベル記述に前記入力データを代入し、前記実行ステップで前記記憶素子が記憶する記憶データを計算するステップと、

演算子演算器データベースに基づいて、計算された前記記憶データの前記実行ステップと前記記憶素子から前記演算子を検索するステップと、

検索された前記演算子の前記出力データと、前記記憶データの異同を判定するステップとをさらに有することを特徴とする請求項 1 または請求項 2 に記載の論理回路設計方法。

【請求項 4】 演算子演算器データベースに基づいて、検索された前記演算子から前記演算器を検索するステップをさらに有することを特徴とする請求項 1 乃至請求項 3 のいずれか 1 つに記載の論理回路設計方法。

【請求項 5】 予想される全実行時間が最短になるようにデータフロー図を変更するステップとをさらに有することを特徴とする請求項 1 乃至請求項 4 のいずれか 1 つに記載の論理回路設計方法。

【請求項 6】 論理回路のアルゴリズムを、複数の演算子を有する動作記述から、前記演算子を実行する演算ノードを実行順に配列したデータフロー図に変換する手順と、

前記データフロー図に実行ステップを割り当て、前記演算ノードからの出力データを記憶するレジスタを前記実行ステップの実行後に挿入する手順と、

前記演算ノードとして機能する演算器と前記レジスタとして機能する記憶素子を有する前記論理回路のデータパスと、前記データパスの制御情報を生成する手順と、

前記演算子を実行する前記演算器を前記演算子から検索可能であり、前記記憶素子の機能をする前記レジスタが記憶するデータを出力する前記演算子を前記実行ステップと前記記憶素子から検索可能である演算子演算器データベースを生成する手順をコンピュータに実行させるための論理回路設計プログラム。

【請求項 7】 前記動作記述と前記データフロー図に基づいて、前記演算ノードで実行する前記演算子を前記演算ノードから検索可能であり、前記レジスタが

記憶するデータを出力する前記演算ノードで実行する前記演算子を前記レジスタから検索可能であるノード演算子データベースを生成する手順と、

前記データフロー図と前記データパスに基づいて、前記演算ノードとして機能する前記演算器を前記演算ノードから検索可能であり、前記記憶素子の機能をする前記レジスタを前記実行ステップと前記記憶素子から検索可能であるノード演算器データベースを生成する手順をさらにコンピュータに実行させるためにあり、

前記演算子演算器データベースを生成する手順が、前記ノード演算子データベースと前記ノード演算器データベースに基づくこととを特徴とする請求項 6 に記載の論理回路設計プログラム。

【請求項 8】 前記データパスと前記制御情報に基づいて、前記論理回路のレジスタ転送レベル記述を生成する手順と、

前記動作記述に入力データを代入し、前記演算子からの出力データを計算する手順と、

前記レジスタ転送レベル記述に前記入力データを代入し、前記実行ステップで前記記憶素子が記憶する記憶データを計算する手順と、

演算子演算器データベースに基づいて、計算された前記記憶データの前記実行ステップと前記記憶素子から前記演算子を検索する手順と、

検索された前記演算子の前記出力データと、前記記憶データの異同を判定する手順をさらにコンピュータに実行させるための請求項 6 または請求項 7 に記載の論理回路設計プログラム。

【請求項 9】 演算子演算器データベースに基づいて、検索された前記演算子から前記演算器を検索する手順をさらにコンピュータに実行させるための請求項 6 乃至請求項 8 のいずれか 1 つに記載の論理回路設計プログラム。

【請求項 10】 予想される全実行時間が最短になるようにデータフロー図を変更する手順をさらにコンピュータに実行させるための請求項 6 乃至請求項 9 のいずれか 1 つに記載の論理回路設計プログラム。

【請求項 11】 論理回路のアルゴリズムを、複数の演算子を有する動作記述から、前記演算子を実行する演算ノードを実行順に配列したデータフロー図に変

換する構文解析部と、

前記データフロー図に実行ステップを割り当て、前記演算ノードからの出力データを記憶するレジスタを前記実行ステップの実行後に挿入するスケジューリング部と、

前記演算ノードとして機能する演算器と前記レジスタとして機能する記憶素子を有する前記論理回路のデータパスと、前記データパスの制御情報を生成するハードウェア割付部と、

前記演算子を実行する前記演算器を前記演算子から検索可能であり、前記記憶素子の機能をする前記レジスタが記憶するデータを出力する前記演算子を前記実行ステップと前記記憶素子から検索可能である演算子演算器データベースを生成する演算子演算器対応情報解析部を有することを特徴とする論理回路設計装置。

【請求項 1 2】 前記動作記述と前記データフロー図に基づいて、前記演算ノードで実行する前記演算子を前記演算ノードから検索可能であり、前記レジスタが記憶するデータを出力する前記演算ノードで実行する前記演算子を前記レジスタから検索可能であるノード演算子データベースを生成するノード演算子対応情報解析部と、

前記データフロー図と前記データパスに基づいて、前記演算ノードとして機能する前記演算器を前記演算ノードから検索可能であり、前記記憶素子の機能をする前記レジスタを前記実行ステップと前記記憶素子から検索可能であるノード演算器データベースを生成するノード演算器対応情報解析部をさらに有し、

前記演算子演算器データベースを生成することが、前記ノード演算子データベースと前記ノード演算器データベースに基づくことを特徴とする請求項 1 1 に記載の論理回路設計装置。

【請求項 1 3】 前記データパスと前記制御情報に基づいて、前記論理回路のレジスタ転送レベル記述を生成するレジスタ転送レベル記述生成部と、

前記動作記述に入力データを代入し、前記演算子からの出力データを計算する動作記述シミュレーション部と、

前記レジスタ転送レベル記述に前記入力データを代入し、前記実行ステップで前記記憶素子が記憶する記憶データを計算するレジスタ転送レベルシミュレーシ

ョン部と、

演算子演算器データベースに基づいて、計算された前記記憶データの前記実行ステップと前記記憶素子から前記演算子を検索する演算子検索部と、

検索された前記演算子の前記出力データと、前記記憶データの異同を判定する記憶データ判定部をさらに有することを特徴とする請求項 1 1 または請求項 1 2 に記載の論理回路設計装置。

【請求項 1 4】 演算子演算器データベースに基づいて、検索された前記演算子から前記演算器を検索する演算器検索部をさらに有することを特徴とする請求項 1 1 乃至請求項 1 3 のいずれか 1 つに記載の論理回路設計装置。

【請求項 1 5】 予想される全実行時間が最短になるようにデータフロー図を変更する最適化部をさらに有することを特徴とする請求項 1 1 乃至請求項 1 4 のいずれか 1 つに記載の論理回路設計装置。

【発明の詳細な説明】

【 0 0 0 1 】

【発明の属する技術分野】

本発明は、レジスタ転送レベル（R T L）記述を自動生成する高位合成技術に関し、特に、動作記述から R T L 記述に変換する論理回路設計方法に関する。

【 0 0 0 2 】

【従来の技術】

従来、論理回路設計では、動作記述の中間信号と論理回路の演算器との対応表が出力された。（例えば、特許文献 1 参照。）。

【 0 0 0 3 】

しかし、さらに高位合成された R T L 記述では、実行ステップ毎にデータが記憶素子に記憶され、1 つの演算器が繰り返し演算を行う。このため、対応表は存在せず、実行ステップ毎の対応表も存在しなかった。R T L 記述の論理回路のシミュレーション結果に不具合があった場合は、設計者が RTL 記述のみを用いてそのシミュレーション結果を解析した。RTL 記述は、動作時間や条件などにより動作する部分が変わるので、この解析には非常に多くの時間を要した。

【 0 0 0 4 】

【特許文献 1】

特開平11-73447号公報（請求項 1、図 6）

【0 0 0 5】

【発明が解決しようとする課題】

本発明は、上記事情に鑑みてなされたものであり、その目的とするところは、動作記述から変換された R T L 記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計方法を提供することにある。

【0 0 0 6】

本発明の目的は、コンピュータが動作記述から変換された R T L 記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計プログラムを提供することにある。

【0 0 0 7】

本発明の目的は、動作記述から変換された R T L 記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計装置を提供することにある。

【0 0 0 8】

【課題を解決するための手段】

上記問題点を解決するための本発明の第 1 の特徴は、論理回路のアルゴリズムを、複数の演算子を有する動作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換することと、

データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入することと、

演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成することと、

演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成することとを有する論理回路設計方法にある。

【0 0 0 9】

本発明の第 2 の特徴は、論理回路のアルゴリズムを、複数の演算子を有する動

作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換する手順と、

データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入する手順と、

演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成する手順と、

演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成する手順をコンピュータに実行させるための論理回路設計プログラムにある。

【 0 0 1 0 】

本発明の第 3 の特徴は、論理回路のアルゴリズムを、複数の演算子を有する動作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換する構文解析部と、

データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入するスケジューリング部と、

演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成するハードウェア割付部と、

演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成する演算子演算器対応情報解析部を有する論理回路設計装置にある。

【 0 0 1 1 】

【発明の実施の形態】

次に、図面を参照して、本発明の実施の形態について説明する。以下の図面の記載において、同一又は類似の部分には同一又は類似の符号を付している。また、図面は模式的なものであり、現実のものとは異なることに留意すべきである。

【 0 0 1 2 】

(論理回路設計装置)

本発明の実施の形態に係る論理回路設計装置 1 は、図 1 に示すように、構文解析部 2、スケジューリング部 3、最適化部 4、ハードウェア割付部 5、RTL 記述生成部 6、ノード演算子対応情報解析部 7、ノード演算器対応情報解析部 8、演算子演算器対応情報解析部 9、動作記述シミュレーション部 10、RTL 記述シミュレーション部 11、シミュレーション結果比較部 12 と入出力部 16 を有している。シミュレーション結果比較部 12 は、演算子検索部 13、記憶データ判定部 14 と演算器検索部 15 を有している。

【0013】

構文解析部 2 は、図 2 に示すように、論理回路のアルゴリズムを、複数の演算子を有する動作記述 D 1 を入力し、演算子を実行する演算ノードを実行順に配列した未加工データフロー図 (DFG) D 2 を出力する。

【0014】

スケジューリング部 3 は、未加工データフロー図 D 2 を入力し、未加工データフロー図 D 2 に実行ステップを割り当てる。スケジューリング部 3 は、演算ノードからの出力データを記憶するレジスタを、未加工データフロー図 D 2 の実行ステップの実行後に挿入する。スケジューリング部 3 は、スケジューリングデータフロー図 D 3 を出力する。

【0015】

最適化部 4 は、スケジューリングデータフロー図 D 3 を入力する。最適化部 4 は、論理回路のアルゴリズムの予想される全実行時間が最短になるように、スケジューリングデータフロー図 D 3 を変更し、最適化データフロー図 D 4 を出力する。

【0016】

ハードウェア割付部 5 は、最適化データフロー図 D 4 を入力する。ハードウェア部 5 は、演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパス D 5 と、データパスの制御情報 D 5 を生成する。

【0017】

RTL 記述生成部 6 は、データパスと制御情報 D 5 を入力する。RTL 記述生

成部 6 は、データパスと制御情報 D 5 に基づいて、論理回路の R T L 記述 D 6 を生成する。R T L 記述生成部 6 は、R T L 記述 D 6 を出力する。

【 0 0 1 8 】

ノード演算子対応情報解析部 7 は、動作記述 D 1、スケジューリングデータフロー図 D 3 と最適化データフロー図 D 4 を入力する。ノード演算子対応情報解析部 7 は、動作記述 D 1、スケジューリングデータフロー図 D 3 と最適化データフロー図 D 4 に基づいて、ノード演算子データベース D 7 を生成する。ノード演算子データベース D 7 は、演算ノードで実行する演算子を演算ノードから検索可能であり、レジスタが記憶するデータを出力する演算ノードで実行する演算子をレジスタから検索可能である。

【 0 0 1 9 】

ノード演算器対応情報解析部 8 は、最適化データフロー図 D 4 とデータパス D 5 を入力する。ノード演算器対応情報解析部 8 は、最適化データフロー図 D 4 とデータパス D 5 に基づいて、ノード演算器データベース D 8 を生成する。ノード演算器データベース D 8 は、演算ノードとして機能する演算器を演算ノードから検索可能であり、記憶素子の機能をするレジスタを実行ステップと記憶素子から検索可能である。

【 0 0 2 0 】

演算子演算器対応情報解析部 9 は、ノード演算子データベース D 7 とノード演算器データベース D 8 を入力する。演算子演算器対応情報解析部 9 は、ノード演算子データベース D 7 とノード演算器データベース D 8 に基づいて、演算子演算器データベース D 9 を生成する。演算子演算器データベース D 9 は、演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である。

【 0 0 2 1 】

動作記述シミュレーション部 1 0 は、動作記述 D 1 と入力データ D 1 3 を入力する。動作記述シミュレーション部 1 0 は、動作記述 D 1 に入力データ D 1 3 を代入し、演算子からの出力データ D 1 0 を計算する。

【 0 0 2 2 】

R T L 記述シミュレーション部 1 1 は、R T L 記述 D 6 と入力データ D 1 3 を入力する。R T L 記述シミュレーション部 1 1 は、R T L 記述 D 6 に入力データ D 1 3 を代入し、実行ステップ毎に記憶素子が記憶する記憶データ D 1 1 を計算する。

【 0 0 2 3 】

シミュレーション結果比較部 1 2 の演算子検索部 1 3 は、演算子演算器データベース D 9 と、記憶データ D 1 1 を記憶した際の実行ステップと記憶素子を入力する。演算子検索部 1 3 は、演算子演算器データベース D 9 に基づいて、計算された記憶データの実行ステップと記憶素子から演算子を検索する。

【 0 0 2 4 】

シミュレーション結果比較部 1 2 の記憶データ判定部 1 4 は、記憶データ D 1 1 と、検索された演算子の出力データを入力する。記憶データ判定部 1 4 は、検索された演算子の出力データと記憶データの異同を判定する。検索された演算子の出力データと記憶データが異なる場合は、R T L 記述 D 6 に不具合があると判定する。この判定に基づいて、R T L 記述 D 6 の不具合の有無 D 1 2 を出力する。R T L 記述 D 6 に不具合がある場合は、さらに、記憶データ D 1 1 を記憶した際の実行ステップ、記憶素子と演算子 D 1 2 を出力する。

【 0 0 2 5 】

シミュレーション結果比較部 1 2 の演算器検索部 1 5 は、検索された演算子の出力データと記憶データが異なる場合に、演算子演算器データベース D 9 と検索された演算子を入力する。演算器検索部 1 5 は、演算子演算器データベース D 9 に基づいて、検索された演算子から演算器を検索する。演算器検索部 1 5 は、検索された演算器 D 1 2 を出力する。

【 0 0 2 6 】

入出力部 1 6 は、動作記述 D 1 と入力データ D 1 3 を入力する。入出力部 1 6 は、R T L 記述 D 6 と R T L 記述 D 6 の不具合の有無、不具合の発生した実行ステップ、記憶素子、演算子と演算器 D 1 2 を出力する。

【 0 0 2 7 】

高位合成で C 記述等の動作記述 D 1 から R T L 記述 D 6 を生成する際に、動作記述

D 1 と RTL 記述 D 6 の両者を対応づける演算子演算器データベース D 9 を生成する。RTL 記述 D 6 の検証に、動作記述 D 1 の検証に用いたテストベクトルを使うことができ、かつ、動作記述 D 1 との比較検証をすることにより、不具合が生じた場合には、動作記述 D 1、RTL 記述 D 6 の両者での不具合箇所が特定できる。このことにより、動作記述 D 1 や RTL 記述 D 6 の修正が容易に行えるようになり、設計、検証、不具合修正にかかる時間を大幅に短縮することができる。

【 0 0 2 8 】

(論理回路設計方法)

本発明の実施の形態に係る論理回路設計方法は、図 3 に示すように、ステップ S 1 において、構文解析部 2 で、構文解析をする。論理回路のアルゴリズムを、複数の演算子を有する動作記述 D 1 から、演算子を実行する演算ノードを実行順に配列した未加工データフロー図 (D F G) D 2 に変換する。

【 0 0 2 9 】

ステップ S 2 において、スケジューリング部 3 で、スケジューリングをする。未加工データフロー図 D 2 に実行ステップを割り当て、演算ノードにどの実行ステップで実行するかという時間情報が添付される。未加工データフロー図 D 2 に、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入する。このレジスタにより、実行ステップをまたがるデータを保持することができる。スケジューリング部 3 は、このように加工された未加工データフロー図 D 2 をスケジューリングデータフロー図 D 3 として出力する。

【 0 0 3 0 】

ステップ S 3 において、最適化部 4 で、予想される全実行時間が最短になるようにスケジューリングデータフロー図 D 3 を変更する。このように変更されたスケジューリングデータフロー図 D 3 を最適化データフロー図 D 4 として出力する。

【 0 0 3 1 】

ステップ S 4 において、ハードウェア割付部 5 で、演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスとこのデータパスの制御情報 D 5 を生成する。

【 0 0 3 2 】

ステップ S 5 において、RTL 記述生成部 6 で、データパスと制御情報 D 5 に基づいて、RTL 記述 D 6 を生成する。

【 0 0 3 3 】

ステップ S 6 において、RTL 記述シミュレーション部 1 1 で、実行ステップ i に、最初に実行される実行ステップ 1 を設定する。

【 0 0 3 4 】

ステップ S 7 において、RTL 記述シミュレーション部 1 1 で、RTL 記述 D 6 のシミュレーションを行う。RTL 記述 D 6 に入力データ D 1 3 を代入し、実行ステップ i で演算器が出力するデータと記憶素子が記憶する記憶データ D 1 1 を計算する。

【 0 0 3 5 】

一方、ステップ S 2 と S 3 の実行後に、ステップ S 8 において、ノード演算子対応情報解析部 7 で、動作記述 D 1 とスケジューリングデータフロー図 D 3 と最適化データフロー図 D 4 に基づいて、ノード演算子データベース D 7 を生成する。

【 0 0 3 6 】

ステップ S 4 の実行後に、ステップ S 9 において、ノード演算器対応情報解析部 8 で、最適化データフロー図 D 4 とデータパス D 5 に基づいて、ノード演算器データベース D 8 を生成する。

【 0 0 3 7 】

ステップ S 1 0 において、演算子演算器対応情報解析部 9 で、ノード演算子データベース D 7 とノード演算器データベース D 8 に基づいて、演算子演算器データベース D 9 を生成する。

【 0 0 3 8 】

ステップ S 1 1 において、動作記述シミュレーション部 1 0 で、動作記述 D 1 のシミュレーションを行う。動作記述 D 1 に入力データ D 1 3 を代入し、演算子からの出力データ D 1 0 を計算する。動作記述を実際にコンパイルし実行した場合の各演算子が出力する中間データを計算する。

【 0 0 3 9 】

ステップ S 7 と S 1 1 の実行後に、ステップ S 1 2 において、シミュレーション結果比較部 1 2 の演算子検索部 1 3 で、演算子演算器データベース D 9 に基づいて、計算された記憶データ D 1 1 の実行ステップ i と記憶素子から演算子を検索する。

【 0 0 4 0 】

ステップ S 1 3 において、記憶データ判定部 1 4 で、検索された演算子の出力データ D 1 0 と、記憶データ D 1 1 の異同を判定する。異なるとの判定の場合は、ステップ S 1 6 に進み、同じとの判定の場合は、ステップ S 1 4 に進む。

【 0 0 4 1 】

ステップ S 1 6 において、演算器検索部 1 5 で、演算子演算器データベース D 9 に基づいて、検索された演算子から演算器を検索する。

【 0 0 4 2 】

ステップ S 1 7 において、論理回路設計装置 1 あるいはそのオペレータが、検索された演算器に基づいて、RTL 記述 D 6 をデバックする。ステップ S 5 に戻る。なお、戻る先は、ステップ S 5 に限らない。再度シミュレーションされる RTL 記述がデバックされた RTL 記述であれば、ステップ S 6 に戻ってもよい。

【 0 0 4 3 】

一方、ステップ S 1 4 において、シミュレーション結果比較部 1 2 で、実行ステップ i が最適化データフロー図 D 4 の最大の実行ステップ以上であるか否かを判定する。実行ステップ i が最大の実行ステップ以上でない場合は、ステップ S 1 5 に進む。ステップ S 1 5 において、実行ステップ i を 1 つ大きくする。そして、ステップ S 7 に進む。実行ステップ i が最大の実行ステップ以上である場合は、論理回路設計方法をストップする。このように、動作記述 D 1 のシミュレーション結果である演算子の出力データ D 1 0 と、RTL 記述 D 6 のシミュレーション結果である記憶素子の記憶データ D 1 1 を、実行ステップ毎に比較することで、動作記述と高位合成された RTL 記述との比較検証が行われる。

【 0 0 4 4 】

動作記述 D 1 と RTL 記述の比較検証をすることにより、不具合が生じた場合に

は、動作記述 D 1、RTL記述 D 6 の両者での不具合箇所が特定できる。このことにより、動作記述 D 1 や RTL記述 D 6 の修正が容易に行えるようになり、設計、検証、不具合修正にかかる時間を大幅に短縮することができる。

【 0 0 4 5 】

尚、RTL記述 D 6 のシミュレーションでは、1実行ステップ毎にデータを比較しつつシミュレーションを行っている。これに限らず、RTL記述 D 6 の全実行ステップの実行後に、動作記述 D 1 と RTL記述の比較検証を行ってもよい。ただ、先の実行ステップでの不具合が、後の実行ステップでの不具合を起こす場合がある一方で、先の実行ステップでの不具合のデバックが、後の実行ステップで不具合を起こす場合もある。従って、動作記述 D 1 と RTL記述の比較検証の時期は、1実行ステップ毎か、全実行ステップの実行後かを、場合により使い分けることが望ましい。

【 0 0 4 6 】

(実施例 1)

実施例 1 では、論理回路設計装置 1 で論理回路設計方法を用いた。実施例 1 の論理回路設計方法は、図 3 のステップ S 1 において、図 4 に示すような複数の演算子 o p 1 乃至 o p 4 を有する動作記述 D 1 から、図 5 に示すような演算子 o p 1 乃至 o p 4 を実行する演算ノード N 1 乃至 N 4 を実行順に配列した未加工データフロー図 D 2 に変換する。演算子 o p 1 乃至 o p 4 は、実際には内部のデータとして管理され、識別番号やポインタ等で認識される。入力データ D 1 3 の引数 i 1 と i 2 で、演算ノード N 1 において、和算の演算子 o p 1 を実行する。引数 i 1 と i 2 で、演算ノード N 2 において、引き算の演算子 o p 3 を実行する。演算ノード N 1 の演算子 o p 1 の出力と演算ノード N 2 の演算子 o p 2 の出力で、演算ノード N 3 において、積算の演算子 o p 2 を実行する。演算ノード N 3 の演算子 o p 2 の出力と定数 1 で、演算ノード N 4 において、和算の演算子 o p 4 を実行する。

【 0 0 4 7 】

ステップ S 2 において、図 6 に示すように、図 5 の未加工データフロー図 D 2 に実行ステップ 1 乃至 3 を割り当てる。演算ノード N 1 と N 2 は、実行ステップ

1で実行するという時間情報が添付される。演算ノードN3は、実行ステップ2で実行するという時間情報が添付される。演算ノードN4は、実行ステップ3で実行するという時間情報が添付される。未加工データフロー図D2に、演算ノードN1乃至N4からの出力データを記憶するレジスタR1乃至R4を実行ステップ1乃至3の実行後に挿入する。スケジューリング部3は、このように加工した図5の未加工データフロー図D2を、図6のスケジューリングデータフロー図D3として出力する。

【0048】

ステップS3において、スケジューリングデータフロー図D3を最適化するが、既に、予想される全実行時間が最短であるので、出力された最適化データフロー図D4は、スケジューリングデータフロー図D3に等しい。

【0049】

ステップS8において、図4の動作記述D1と図6のスケジューリングデータフロー図D3に基づいて、図7に示すようなノード演算子データベースD7を生成する。ノード演算子データベースD7は、複数のノード演算子レコード21を有している。ノード演算子レコード21は、ノードフィールド22と演算子フィールド23を有している。ノードフィールド22では、演算ノードN1乃至N4とレジスタR1乃至R4が記憶される。演算子フィールド23では、演算子op1乃至op4が記憶される。演算子op1乃至op4とその演算子op1乃至op4を実行する演算ノードN1乃至N4が、同じノード演算子レコード21内に記憶される。また、演算子op1乃至op4とその演算子op1乃至op4の出力データを記憶するレジスタR1乃至R4が、同じノード演算子レコード21内に記憶される。演算ノードN1乃至N4又はレジスタR1乃至R4から、同じノード演算子レコード21内に記憶された演算子op1乃至op4を検索することが可能である。

【0050】

ステップS4において、ハードウェア割付部5で、図8に示すような論理回路のデータパスD5と、このデータパスの制御情報D5を生成する。データパスD5は、演算ノードN1乃至N4として機能する演算器A1乃至A3と、レジスタ

R 1 乃至 R 4 として機能する記憶素子 M 1 と M 2 と、制御情報 D 5 に基づいて伝達するデータを切り換えるセレクター C 1 乃至 C 3 を有する。

【 0 0 5 1 】

ステップ S 9 において、図 6 のスケジューリングデータフロー図 D 3 と図 8 のデータパス D 5 に基づいて、図 9 に示すようなノード演算器データベース D 8 を生成する。ノード演算器データベース D 8 は、複数のノード演算器レコード 2 4 を有している。ノード演算器レコード 2 4 は、演算器フィールド 2 5、ノードフィールド 2 6 と実行ステップフィールド 2 7 を有している。演算器フィールド 2 5 では、演算器 A 1 乃至 A 3 と記憶素子 M 1、M 2 が記憶される。ノードフィールド 2 2 では、演算ノード N 1 乃至 N 4 とレジスタ R 1 乃至 R 4 が記憶される。実行ステップフィールド 2 7 では、実行ステップ 1 乃至 3 が記憶される。演算器 A 1 乃至 A 3 と、その演算器 A 1 乃至 A 3 で実行する演算ノード N 1 乃至 N 4 と、その演算ノード N 1 乃至 N 4 が実行する実行ステップが、同じノード演算器レコード 2 4 内に記憶される。また、記憶素子 M 1、M 2 と、その記憶素子 M 1、M 2 で記憶するレジスタ R 1 乃至 R 4 と、そのレジスタ R 1 乃至 R 4 が記憶する実行ステップが、同じノード演算器レコード 2 4 内に記憶される。

【 0 0 5 2 】

すなわち、図 6 のスケジューリングデータフロー図 D 3 と図 8 のデータパス D 5 を参照しながら、図 9 のノード演算器データベース D 8 を見ると、実行ステップ 1 において、演算器 A 1 が演算ノード N 1 として機能し、演算器 A 2 が演算ノード N 2 として機能し、演算器 A 3 は働かず、記憶素子 M 1 がレジスタ R 1 として機能し、記憶素子 M 2 がレジスタ R 2 として機能する。実行ステップ 2 において、演算器 A 1 と演算器 A 2 は働かず、演算器 A 3 が演算ノード N 3 として機能し、記憶素子 M 1 がレジスタ R 3 として機能し、記憶素子 M 2 は働かない。実行ステップ 3 において、演算器 A 1 が演算ノード N 4 として機能し、演算器 A 2 と演算器 A 3 は働かず、記憶素子 M 1 がレジスタ R 4 として機能し、記憶素子 M 2 は働かない。

【 0 0 5 3 】

図 9 のノード演算器データベース D 8 によれば、演算ノード N 1 乃至 N 4 から

、同じノード演算器レコード 2 4 内に記憶された演算器 A 1 乃至 A 3 を検索することが可能である。記憶素子 M 1、M 2 と実行ステップ 1 乃至 3 から同じノード演算器レコード 2 4 内に記憶されたレジスタ R 1 乃至 R 4 を検索することが可能である。

【 0 0 5 4 】

ステップ S 1 0 において、図 7 のノード演算子データベース D 7 と図 9 のノード演算器データベース D 8 に基づいて、図 1 0 に示すような演算子演算器データベース D 9 を生成する。演算子演算器データベース D 9 は、複数の演算子演算器レコード 2 8 を有している。演算子演算器レコード 2 8 は、演算器フィールド 2 9、演算子フィールド 3 0 と実行ステップフィールド 3 1 を有している。演算器フィールド 2 9 では、演算器 A 1 乃至 A 3 と記憶素子 M 1、M 2 が記憶される。演算子フィールド 3 0 では、演算子 o p 1 乃至 o p 4 が記憶される。実行ステップフィールド 3 1 では、実行ステップ 1 乃至 3 が記憶される。演算器 A 1 乃至 A 3 と、その演算器 A 1 乃至 A 3 で実行する演算子 o p 1 乃至 o p 4 と、その演算子 o p 1 乃至 o p 4 が実行する実行ステップが、同じ演算子演算器レコード 2 8 内に記憶される。また、記憶素子 M 1、M 2 と、その記憶素子 M 1、M 2 で記憶するデータを出力する演算子 o p 1 乃至 o p 4 と、その演算子 o p 1 乃至 o p 4 が実行する実行ステップが、同じ演算子演算器レコード 2 8 内に記憶される。

【 0 0 5 5 】

すなわち、図 7 のノード演算子データベース D 7 と図 9 のノード演算器データベース D 8 を参照しながら、図 1 0 の演算子演算器データベース D 9 を見ると、実行ステップ 1 において、演算器 A 1 が演算子 o p 1 として機能し、演算器 A 2 が演算子 o p 3 として機能し、演算器 A 3 は働かず、記憶素子 M 1 が演算子 o p 1 の出力したデータを記憶し、記憶素子 M 2 が演算子 o p 3 の出力したデータを記憶する。実行ステップ 2 において、演算器 A 1 と演算器 A 2 は働かず、演算器 A 3 が演算子 o p 2 として機能し、記憶素子 M 1 が演算子 o p 2 の出力したデータを記憶し、記憶素子 M 2 は働かない。実行ステップ 3 において、演算器 A 1 が演算子 o p 4 として機能し、演算器 A 2 と演算器 A 3 は働かず、記憶素子 M 1 が演算子 o p 4 の出力したデータを記憶し、記憶素子 M 2 は働かない。

【 0 0 5 6 】

図 1 0 の演算子演算器データベース D 9 によれば、演算子 o p 1 乃至 o p 4 から、同じ演算子演算器レコード 2 8 内に記憶された演算器 A 1 乃至 A 3 を検索することが可能である。記憶素子 M 1、M 2 と実行ステップ 1 乃至 3 から同じ演算子演算器レコード 2 8 内に記憶された演算子 o p 1 乃至 o p 4 を検索することが可能である。

【 0 0 5 7 】

ステップ S 1 1 において、動作記述 D 1 のシミュレーションを行う。図 1 1 に示すように、入力データ D 1 3 として、図 4 の動作記述 D 1 の引数 i 1 に 2 を代入し、引数 i 2 に 1 を代入した。そして、演算子 o p 1 からの出力データ D 1 0 として 3 を出力した。同様に、演算子 o p 2、o p 3、o p 4 からの出力データ D 1 0 として 3、1、4 を出力した。

【 0 0 5 8 】

次に、動作記述 D 1 の引数 i 1 に 3 を代入し、引数 i 2 に 2 を代入した。そして、演算子 o p 1、o p 2、o p 3、o p 4 からの出力データ D 1 0 として 5、5、1、6 を出力した。さらに、動作記述 D 1 の引数 i 1 に 2 を代入し、引数 i 2 に 3 を代入した。そして、演算子 o p 1、o p 2、o p 3、o p 4 からの出力データ D 1 0 として 5、- 5、- 1、- 4 を出力した。

【 0 0 5 9 】

なお、入力データ D 1 3 と出力データ D 1 0 は、図 1 1 に示すように、それぞれフィールドを構成し、入力データ D 1 3 と出力データ D 1 0 と演算子フィールド 3 3 からなる入力データ出力データレコード 3 2 を有するデータベースを構成してもよい。入力データ出力データレコード 3 2 を有するデータベースによれば、演算子 o p 1 乃至 o p 4 と入力データ D 1 3 から、同じ入力データ出力データレコード 3 2 内に記憶された出力データ D 1 0 を検索することが可能である。

【 0 0 6 0 】

ステップ S 5 において、R T L 記述生成部 6 で、データパスと制御情報 D 5 に基づいて、R T L 記述 D 6 を生成する。ステップ S 6 において、R T L 記述シミュレーション部 1 1 で、実行ステップ i に、最初に実行される実行ステップ 1 を

設定する。

【 0 0 6 1 】

ステップ S 7 において、入力データ D 1 3 で引数 i 1 が 2 であり、引数 i 2 が 1 である場合の RTL 記述 D 6 のシミュレーションを行う。図 1 2 に示すように、まず、実行ステップ 1 で、RTL 記述 D 6 に入力データ D 1 3 を代入し、記憶素子 M 1、M 2 が記憶する記憶データ D 1 1 を計算する。RTL 記述 D 6 の引数 i 1 に 2 を代入し、引数 i 2 に 1 を代入した。そして、実行ステップ 1 で、記憶素子 M 1 の記憶データ D 1 1 として 3 を出力し、記憶素子 M 2 の記憶データ D 1 1 として 1 を出力した。

【 0 0 6 2 】

ステップ S 1 2 において、図 1 0 の演算子演算器データベース D 9 に基づいて、計算された記憶データ D 1 1 の実行ステップ 1 と記憶素子 M 1 から演算子 o p 1 を検索する。また、実行ステップ 1 と記憶素子 M 2 から演算子 o p 3 を検索する。

【 0 0 6 3 】

ステップ S 1 3 において、図 1 1 の検索された演算子 o p 1 の出力データ D 1 0 である 3 と、図 1 2 の実行ステップ 1 で記憶素子 M 1 の記憶データ D 1 1 である 3 の異同を判定する。3 と 3 で同じである。また、図 1 1 の検索された演算子 o p 3 の出力データ D 1 0 である 1 と、図 1 2 の実行ステップ 1 で記憶素子 M 2 の記憶データ D 1 1 である 1 の異同を判定する。1 と 1 で同じである。これらにより、ステップ S 1 4 に進む。

【 0 0 6 4 】

ステップ S 1 4 において、実行ステップ 1 が最適化データフロー図 D 4 の最大の実行ステップ 3 以上であるか否かを判定する。実行ステップ 1 が最大の実行ステップ 3 以上でないので、ステップ S 1 5 に進む。ステップ S 1 5 において、実行ステップ 1 を 1 つ大きく実行ステップ 2 にする。そして、ステップ S 7 に進む。

【 0 0 6 5 】

再度、ステップ S 7 において、入力データ D 1 3 で引数 i 1 が 2 であり、引数

i 2 が 1 である場合で実行ステップ 2 の RTL 記述 D 6 のシミュレーションを行う。
図 1 2 に示すように、実行ステップ 2 で、記憶素子 M 1 の記憶データ D 1 1 と
して 3 を出力した。

【 0 0 6 6 】

ステップ S 1 2 において、図 1 0 の演算子演算器データベース D 9 に基づいて、
実行ステップ 2 と記憶素子 M 1 から演算子 o p 2 を検索する。

【 0 0 6 7 】

ステップ S 1 3 において、図 1 1 の検索された演算子 o p 2 の出力データ D 1
0 である 3 と、図 1 2 の実行ステップ 2 で記憶素子 M 1 の記憶データ D 1 1 であ
る 3 の異同を判定する。3 と 3 で同じであるので、ステップ S 1 4 に進む。

【 0 0 6 8 】

ステップ S 1 4 において、実行ステップ 2 が最適化データフロー図 D 4 の最大
の実行ステップ 3 以上であるか否かを判定する。実行ステップ 2 が最大の実行ス
テップ 3 以上でないので、ステップ S 1 5 に進む。ステップ S 1 5 において、実
行ステップ 2 を 1 つ大きく実行ステップ 3 にする。そして、ステップ S 7 に進む
。

【 0 0 6 9 】

三度、ステップ S 7 において、入力データ D 1 3 で引数 i 1 が 2 であり、引数 i
2 が 1 である場合で実行ステップ 3 の RTL 記述 D 6 のシミュレーションを行う。
図 1 2 に示すように、実行ステップ 3 で、記憶素子 M 1 の記憶データ D 1 1 とし
て 4 を出力した。

【 0 0 7 0 】

ステップ S 1 2 において、図 1 0 の演算子演算器データベース D 9 に基づいて、
実行ステップ 3 と記憶素子 M 1 から演算子 o p 4 を検索する。

【 0 0 7 1 】

ステップ S 1 3 において、図 1 1 の検索された演算子 o p 4 の出力データ D 1
0 である 4 と、図 1 2 の実行ステップ 3 で記憶素子 M 1 の記憶データ D 1 1 であ
る 4 の異同を判定する。4 と 4 で同じであるので、ステップ S 1 4 に進む。

【 0 0 7 2 】

ステップ S 1 4 において、実行ステップ 3 が最適化データフロー図 D 4 の最大の実行ステップ 3 以上であるか否かを判定する。実行ステップ 3 が最大の実行ステップ 3 以上であるので、論理回路設計方法をストップする。

【 0 0 7 3 】

このように、動作記述 D 1 のミュレーション結果である演算子の出力データ D 1 1 0 と、RTL記述 D 6 のシミュレーション結果である記憶素子の記憶データ D 1 1 が、実行ステップ毎に一致することを確認することで、動作記述からRTL記述が正確に高位合成されたことが検証できた。

【 0 0 7 4 】

(実施例 2)

RTL記述 D 6 のシミュレーションを、1実行ステップ毎にデータを比較をするのではなく、RTL記述 D 6 の全実行ステップの実行後に動作記述 D 1 とRTL記述の比較検証を行う場合について説明する。入力データ D 1 3 は、引数 i 1 が 3 であり、引数 i 2 が 2 であるとする。実施例 1 とステップ S 1 乃至 S 5、S 8 乃至 S 1 1 は同じである。ステップ S 1 2、S 1 3、S 1 6、S 1 7 の実施に先駆けて、ステップ S 6、S 7、S 1 4 と S 1 5 を実施する。ステップ S 7、S 1 4 と S 1 5 でループを構成する。実行ステップ 1 から最大の実行ステップ 3 までの実行ステップ i について、入力データ D 1 3 で引数 i 1 が 3 であり、引数 i 2 が 2 である場合のRTL記述 D 6 のシミュレーションを行う。図 1 2 に示すように、まず、実行ステップ 1 で、記憶素子 M 1 の記憶データ D 1 1 として 5 を出力し、記憶素子 M 2 の記憶データ D 1 1 として 1 を出力した。次に、実行ステップ 2 で、記憶素子 M 1 の記憶データ D 1 1 として 5 を出力した。最後に、実行ステップ 3 で、記憶素子 M 1 の記憶データ D 1 1 として 6 を出力した。

【 0 0 7 5 】

ステップ S 1 2 において、図 1 0 の演算子演算器データベース D 9 に基づいて、計算された記憶データ D 1 1 の実行ステップ 1 と記憶素子 M 1 から演算子 o p 1 を検索する。また、実行ステップ 1 と記憶素子 M 2 から演算子 o p 3 を検索する。実行ステップ 2 と記憶素子 M 1 から演算子 o p 2 を検索する。実行ステップ 3 と記憶素子 M 1 から演算子 o p 4 を検索する。

【 0 0 7 6 】

ステップ S 1 3 において、図 1 1 の検索された演算子 o p 1 の出力データ D 1 0 である 5 と、図 1 2 の実行ステップ 1 で記憶素子 M 1 の記憶データ D 1 1 である 5 の異同を判定する。5 と 5 で同じである。また、図 1 1 の検索された演算子 o p 3 の出力データ D 1 0 である 1 と、図 1 2 の実行ステップ 1 で記憶素子 M 2 の記憶データ D 1 1 である 1 の異同を判定する。1 と 1 で同じである。図 1 1 の検索された演算子 o p 2 の出力データ D 1 0 である 5 と、図 1 2 の実行ステップ 2 で記憶素子 M 1 の記憶データ D 1 1 である 5 の異同を判定する。5 と 5 で同じである。図 1 1 の検索された演算子 o p 4 の出力データ D 1 0 である 6 と、図 1 2 の実行ステップ 3 で記憶素子 M 1 の記憶データ D 1 1 である 6 の異同を判定する。6 と 6 で同じである。これらの判定により、論理回路設計方法をストップする。

【 0 0 7 7 】

このように、動作記述 D 1 のミュレーション結果である演算子の出力データ D 1 0 と、RTL記述 D 6 のシミュレーション結果である記憶素子の記憶データ D 1 1 を、全実行ステップにわたって一致することを確認することで、動作記述から RTL記述が正確に高位合成されたことが検証できた。

【 0 0 7 8 】

(実施例 3)

RTL記述 D 6 に不具合がある場合について説明する。RTL記述 D 6 のシミュレーションは、実施例 1 と同様に、RTL記述 D 6 の全実行ステップの実行後に動作記述 D 1 と RTL記述 D 6 の比較検証を行った。入力データ D 1 3 は、引数 i 1 が 2 であり、引数 i 2 が 3 であるとする。実施例 1 とステップ S 1 乃至 S 5、S 8 乃至 S 1 1 は同じである。実施例 2 と同様に、ステップ S 1 2、S 1 3、S 1 6、S 1 7 の実施に先駆けて、ステップ S 6、S 7、S 1 4 と S 1 5 を実施する。図 1 2 に示すように、まず、実行ステップ 1 で、記憶素子 M 1 の記憶データ D 1 1 として 5 を出力し、記憶素子 M 2 の記憶データ D 1 1 として - 1 を出力した。次に、実行ステップ 2 で、記憶素子 M 1 の記憶データ D 1 1 として 1 2 7 5 を出力した。最後に、実行ステップ 3 で、記憶素子 M 1 の記憶データ D 1 1 として 1 2

7 6 を出力した。

【 0 0 7 9 】

ステップ S 1 2 において、図 1 0 の演算子演算器データベース D 9 に基づいて、計算された記憶データ D 1 1 の実行ステップ 1 と記憶素子 M 1 から演算子 o p 1 を検索する。また、実行ステップ 1 と記憶素子 M 2 から演算子 o p 3 を検索する。実行ステップ 2 と記憶素子 M 1 から演算子 o p 2 を検索する。実行ステップ 3 と記憶素子 M 1 から演算子 o p 4 を検索する。

【 0 0 8 0 】

ステップ S 1 3 において、図 1 1 の検索された演算子 o p 1 の出力データ D 1 0 である 5 と、図 1 2 の実行ステップ 1 で記憶素子 M 1 の記憶データ D 1 1 である 5 の異同を判定する。5 と 5 で同じである。また、図 1 1 の検索された演算子 o p 3 の出力データ D 1 0 である - 1 と、図 1 2 の実行ステップ 1 で記憶素子 M 2 の記憶データ D 1 1 である - 1 の異同を判定する。- 1 と - 1 で同じである。図 1 1 の検索された演算子 o p 2 の出力データ D 1 0 である - 5 と、図 1 2 の実行ステップ 2 で記憶素子 M 1 の記憶データ D 1 1 である 1 2 7 5 の異同を判定する。- 5 と 1 2 7 5 は異なる。図 1 1 の検索された演算子 o p 4 の出力データ D 1 0 である - 4 と、図 1 2 の実行ステップ 3 で記憶素子 M 1 の記憶データ D 1 1 である 1 2 7 6 の異同を判定する。- 4 と 1 2 7 6 は異なる。異なる判定が生じたのでステップ S 1 6 に進む。検索された演算子 o p 2 と o p 4 で異なる判定が生じた。

【 0 0 8 1 】

ステップ S 1 6 において、図 1 0 の演算子演算器データベース D 9 に基づいて、異なる判定が生じた演算子 o p 2 から演算器 A 3 を検索する。同様に、異なる判定が生じた演算子 o p 4 から演算器 A 1 を検索する。

【 0 0 8 2 】

ステップ S 1 7 において、論理回路設計装置 1 のオペレータが、検索された演算器 A 3 と A 1 に基づいて、R T L 記述 D 6 をデバックする。なお、演算器 A 3 は演算器 A 1 より早い実行ステップ 2 で不具合を発生させている。演算器 A 3 での不具合により、演算器 A 1 の不具合が発生する場合も考えられる。このような

場合は、まず、演算器 A 3 に関してデバックを行えばよい。そして、ステップ S 5 に戻る。ステップ S 5 に戻ることで、デバックされた R T L 記述 D 6 の記憶素子の記憶データ D 1 1 の値が、動作記述 D 1 の演算子の出力データ D 1 0 に一致するまで、オペレータはデバックを繰り返すことができる。

【 0 0 8 3 】

このように、動作記述 D 1 と R T L 記述 D 6 の比較検証をすることにより、不具合が生じた場合には、動作記述 D 1 と R T L 記述 D 6 の両者での不具合箇所が特定できる。このことにより、R T L 記述 D 6 の修正が容易に行え、設計にかかる時間を大幅に短縮することができる。

【 0 0 8 4 】

(実施例 4)

実施例 4 でも、論理回路設計装置 1 で論理回路設計方法を用いた。実施例 4 では最適化を実施した場合について説明する。実施例 4 の論理回路設計方法は、図 3 のステップ S 1 において、図 1 3 に示すような複数の演算子 o p 1 乃至 o p 3 を有する動作記述 D 1 から、演算子 o p 1 乃至 o p 3 を実行する演算ノード N 1 乃至 N 3 を実行順に配列した未加工データフロー図 D 2 に変換する。

【 0 0 8 5 】

ステップ S 2 において、図 1 4 に示すように、未加工データフロー図 D 2 に実行ステップ 1 乃至 3 を割り当てる。入力データ D 1 3 の引数 i 1 と i 2 で、演算ノード N 1 において、和算の演算子 o p 1 を実行する。演算ノード N 1 は、実行ステップ 1 で実行するという時間情報が添付される。演算ノード N 1 からの出力データを記憶するレジスタ R 1 を実行ステップ 1 の実行後に挿入する。

【 0 0 8 6 】

演算ノード N 1 の演算子 o p 1 の出力と引数 i 3 で、演算ノード N 2 において、和算の演算子 o p 2 を実行する。演算ノード N 2 は、実行ステップ 2 で実行するという時間情報が添付される。演算ノード N 2 からの出力データを記憶するレジスタ R 2 を実行ステップ 2 の実行後に挿入する。

【 0 0 8 7 】

演算ノード N 2 の演算子 o p 2 の出力と引数 i 4 で、演算ノード N 3 において

、和算の演算子 op_3 を実行する。演算ノード N_3 は、実行ステップ 3 で実行するという時間情報が添付される。演算ノード N_3 からの出力データを記憶するレジスタ R_3 を実行ステップ 3 の実行後に挿入する。

【 0 0 8 8 】

スケジューリング部 3 は、このように加工した未加工データフロー図 D_2 を、図 1 4 のスケジューリングデータフロー図 D_3 として出力する。なお、後述するステップ S_8 の理解が容易になるように、ステップ S_8 の一部を先行して実施する。図 1 3 の動作記述 D_1 と図 1 4 のスケジューリングデータフロー図 D_3 に基づいて、図 1 5 に示すようなノード演算子データベース D_7 を生成する。ノードフィールド 2 2 では、演算ノード N_1 乃至 N_3 とレジスタ R_1 乃至 R_3 が記憶される。演算子フィールド 2 3 では、演算子 op_1 乃至 op_3 が記憶される。演算子 op_1 乃至 op_3 とその演算子 op_1 乃至 op_3 を実行する演算ノード N_1 乃至 N_3 が、同じノード演算子レコード 2 1 内に記憶される。また、演算子 op_1 乃至 op_3 とその演算子 op_1 乃至 op_3 の出力データを記憶するレジスタ R_1 乃至 R_3 が、同じノード演算子レコード 2 1 内に記憶される。

【 0 0 8 9 】

ステップ S_3 において、図 1 4 のスケジューリングデータフロー図 D_3 を、図 1 6 に示すように最適化し、最適化データフロー図 D_4 を生成した。直列に処理される和算を並列に処理するように組み替えた。演算ノード N_2 、 N_3 とレジスタ R_2 が消去されて、代わりに演算ノード N_4 、 N_5 とレジスタ R_4 が生成された。予想される全実行時間が、実行ステップで 3 実行ステップから 2 実行ステップに短縮された。

【 0 0 9 0 】

ステップ S_8 において、図 1 3 の動作記述 D_1 と図 1 4 のスケジューリングデータフロー図 D_3 と図 1 6 の最適化データフロー図 D_4 に基づいて、まず、図 1 5 のノード演算子データベース D_7 を生成し、さらに、図 1 7 に示すような、ノード演算子データベース D_7 を生成する。

【 0 0 9 1 】

ノードフィールド 2 2 では、演算ノード N_1 乃至 N_5 とレジスタ R_1 乃至 R_4

が記憶される。演算子フィールド 2 3 では、演算子 o p 1 乃至 o p 3 が記憶される。演算子 o p 1 乃至 o p 3 とその演算子 o p 1 乃至 o p 3 を実行する演算ノード N 1 乃至 N 4 が、同じノード演算子レコード 2 1 内に記憶される。なお、演算子 o p 2 と o p 3 に直接対応する演算ノード N 2 と N 3 は、図 1 6 の最適化データフロー図 D 4 では存在しない。そこで、演算子 o p 2 と o p 3 の実行に影響を受ける演算ノードに、演算ノード N 4 と N 5 を設定する。演算子 o p 2 と o p 3 とその演算子 o p 2 と o p 3 の実行に影響を受ける演算ノード N 4 が、同じノード演算子レコード 2 1 内に記憶される。演算子 o p 2 と o p 3 とその演算子 o p 2 と o p 3 の実行に影響を受ける演算ノード N 5 が、同じノード演算子レコード 2 1 内に記憶される。

【 0 0 9 2 】

また、演算ノード N 1 と、演算ノード N 1 に対応するレジスタ R 1 は、演算子 o p 1 に関係付けられている。演算ノード N 4 と、演算ノード N 4 に対応するレジスタ R 4 は、演算子 o p 2、o p 3 に関係付けられている。演算ノード N 5 と、演算ノード N 5 に対応するレジスタ R 3 は、演算子 o p 2、o p 3 に関係付けられている。

【 0 0 9 3 】

ステップ S 4 において、図 1 8 に示すような論理回路のデータパス D 5 と、このデータパスの制御情報 D 5 を生成する。データパス D 5 は、演算ノード N 1、N 5 として機能する演算器 A 1 と、演算ノード N 4 として機能する演算器 A 2 と、レジスタ R 1、R 3 として機能する記憶素子 M 1 と、レジスタ R 4 として機能する記憶素子 M 2 と、制御情報 D 5 に基づいて伝達するデータを切り換えるセレクター C 1、C 2 を有する。

【 0 0 9 4 】

ステップ S 9 において、図 1 6 の最適化データフロー図 D 4 と図 1 8 のデータパス D 5 に基づいて、図 1 9 に示すようなノード演算器データベース D 8 を生成する。図 1 9 のノード演算器データベース D 8 より、実行ステップ 1 において、演算器 A 1 が演算ノード N 1 として機能し、演算器 A 2 が演算ノード N 4 として機能し、記憶素子 M 1 がレジスタ R 1 として機能し、記憶素子 M 2 がレジスタ R

4として機能する。実行ステップ2において、演算器A1が演算ノードN5として機能し、記憶素子M1がレジスタR3として機能する。

【0095】

ステップS10において、図17のノード演算子データベースD7と図19のノード演算器データベースD8に基づいて、図20に示すような演算子演算器データベースD9を生成する。図20の演算子演算器データベースD9を見ると、実行ステップ1において、演算器A1が演算子op1として機能し、演算器A2が演算子op2、op3として機能し、記憶素子M1が演算子op1の出力したデータを記憶し、記憶素子M2が演算子op2、op3の出力したデータを記憶する。実行ステップ2において、演算器A1が演算子op2、op3として機能し、記憶素子M1が演算子op3の出力したデータを記憶する。

【0096】

図20の演算子演算器データベースD9によれば、演算子op1乃至op3から、同じ演算子演算器レコード28内に記憶された演算器A1、A2を検索することが可能である。記憶素子M1、M2と実行ステップ1、2から同じ演算子演算器レコード28内に記憶された演算子op1乃至op3を検索することが可能である。例えば、記憶素子M1と実行ステップ1から演算子op1を検索できる。そして、演算子op1から演算器A1を検索できる。また、記憶素子M2と実行ステップ1から演算子op2、op3を検索できる。演算子op2、op3と実行ステップ1から演算器A2を検索できる。

【0097】

このように、動作記述D1からRTL記述D6を生成する際に、動作記述D1とRTL記述D6の両者を対応づける演算子演算器データベースD9が生成できる。このことにより、動作記述D1とRTL記述D6の比較検証をすることができる。この比較検証で不具合が生じた場合には、動作記述D1、RTL記述D6の両者での不具合箇所が特定できる。動作記述D1やRTL記述D6の修正が容易に行える。

【0098】

【発明の効果】

以上説明したように、本発明によれば、動作記述から変換されたRTL記述で

の論理回路のシミュレーション結果を迅速に解析するための論理回路設計方法を提供できる。

【 0 0 9 9 】

本発明によれば、コンピュータが動作記述から変換された R T L 記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計プログラムを提供できる。

【 0 1 0 0 】

本発明によれば、動作記述から変換された R T L 記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計装置を提供できる。

【図面の簡単な説明】

【図 1】

本発明の実施の形態に係る論理回路設計装置の構成図である。

【図 2】

本発明の実施の形態に係る論理回路設計装置のデータフロー図である。

【図 3】

本発明の実施の形態に係る論理回路設計方法のフローチャートである。

【図 4】

実施例 1 の動作記述を表す図である。

【図 5】

実施例 1 の未加工の D F G を表す図である。

【図 6】

実施例 1 のスケジューリング後の D F G を表す図である。

【図 7】

実施例 1 のノード演算子データベースのデータ構造を表す図である。

【図 8】

実施例 1 のデータパスを表す図である。

【図 9】

実施例 1 のノード演算器データベースのデータ構造を表す図である。

【図 1 0】

実施例 1 の演算子演算器データベースのデータ構造を表す図である。

【図 1 1】

実施例 1 の動作記述のシミュレーションの結果を表す図である。

【図 1 2】

実施例 1 の R T L 記述のシミュレーションの結果を表す図である。

【図 1 3】

実施例 4 の動作記述を表す図である。

【図 1 4】

実施例 4 のスケジューリング後の D F G を表す図である。

【図 1 5】

実施例 4 のノード演算子データベースの最適化前のデータ構造を表す図である

【図 1 6】

実施例 4 の最適化後の D F G を表す図である。

【図 1 7】

実施例 4 のノード演算子データベースの最適化後のデータ構造を表す図である

【図 1 8】

実施例 4 のデータパスを表す図である。

【図 1 9】

実施例 4 のノード演算器データベースのデータ構造を表す図である。

【図 2 0】

実施例 4 の演算子演算器データベースのデータ構造を表す図である。

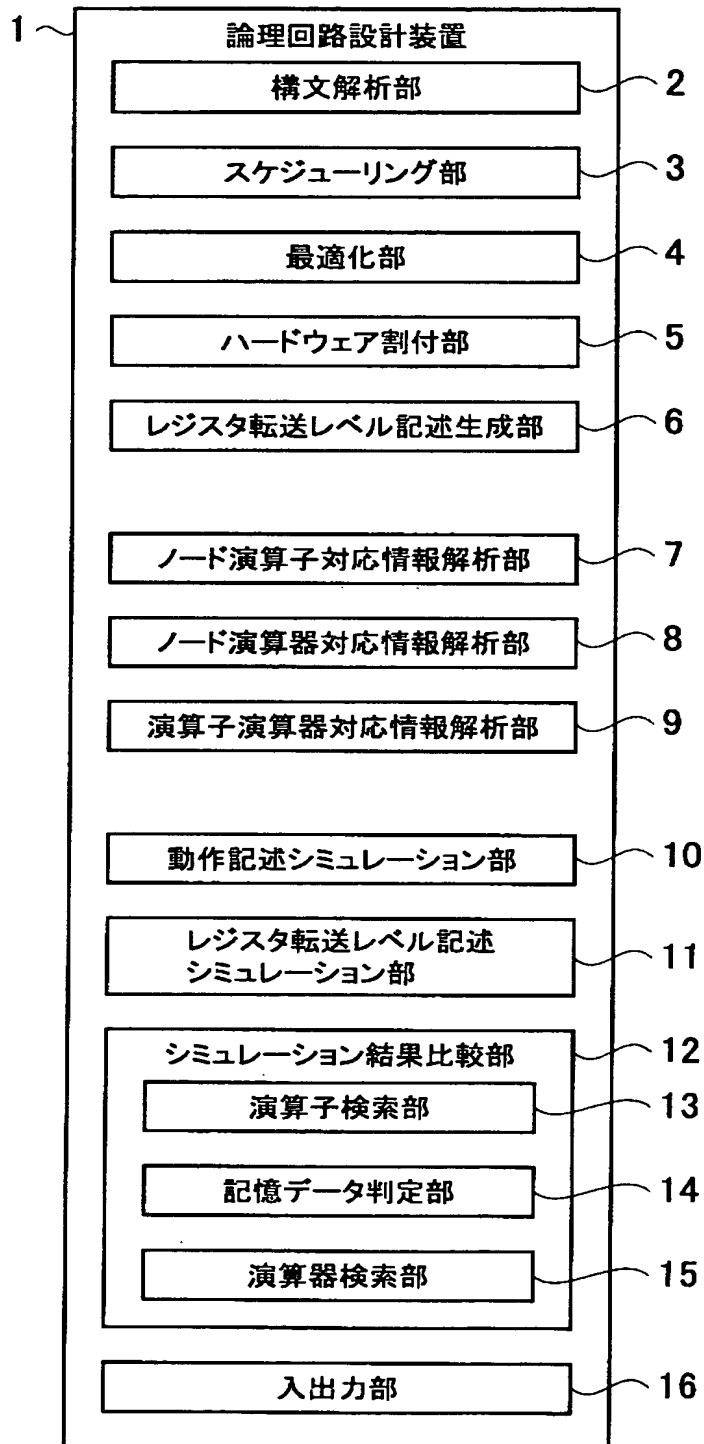
【符号の説明】

- 1 論理回路設計装置
- 2 構文解析部
- 3 スケジューリング部
- 4 最適化部
- 5 ハードウェア割付部

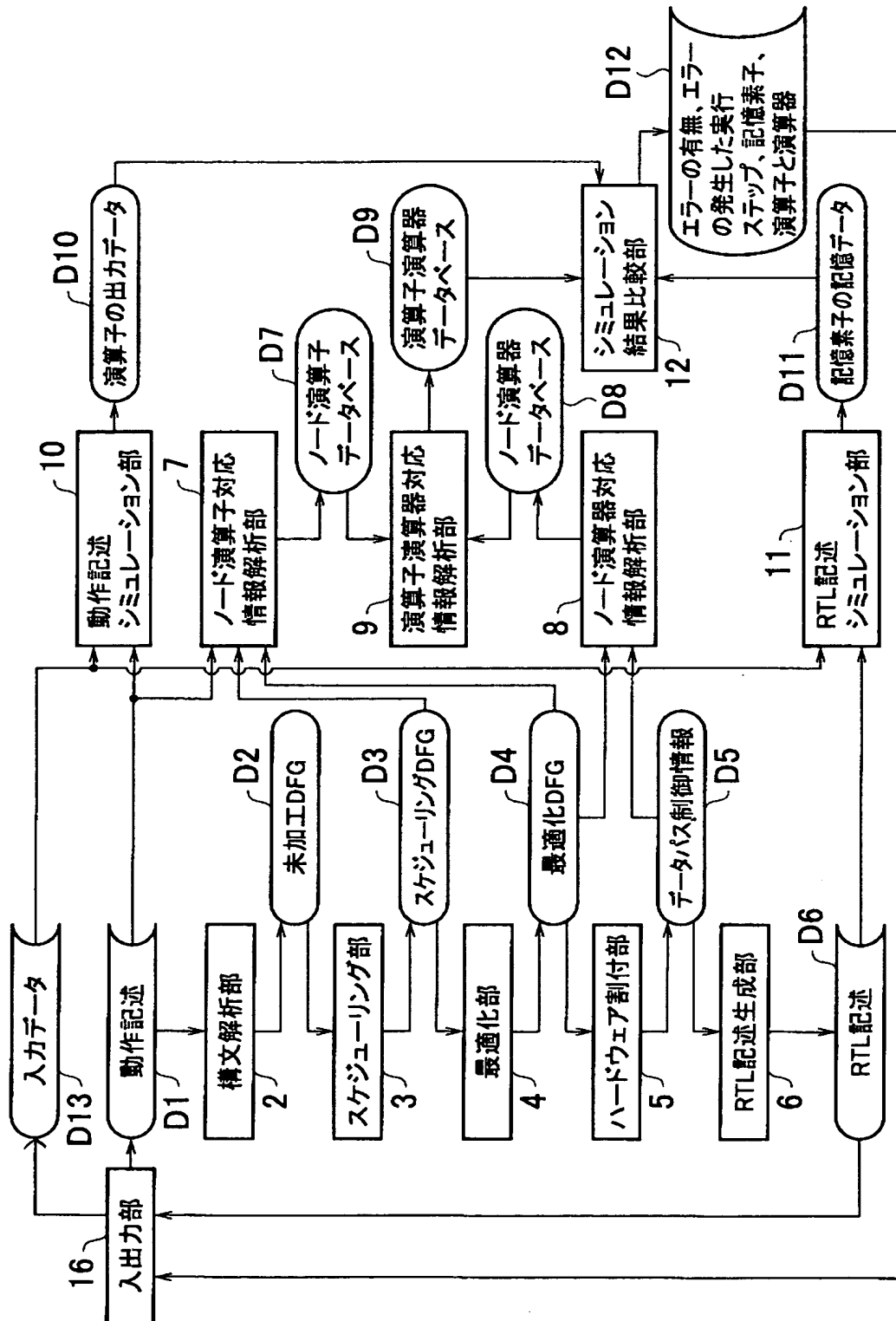
- 6 レジスタ転送レベル（R T L）記述生成部
- 7 ノード演算子対応情報解析部
- 8 ノード演算器対応情報解析部
- 9 演算子演算器対応情報解析部
- 1 0 動作記述シミュレーション部
- 1 1 R T L 記述シミュレーション部
- 1 2 シミュレーション結果比較部
- 1 3 演算子検索部
- 1 4 記憶データ判定部
- 1 5 演算器検索部
- 1 6 入出力部
- 2 1 ノード演算子レコード
- 2 2 ノードフィールド
- 2 3 演算子フィールド
- 2 4 ノード演算器レコード
- 2 5 演算器フィールド
- 2 6 ノードフィールド
- 2 7 実行ステップフィールド
- 2 8 演算子演算器レコード
- 2 9 演算器フィールド
- 3 0 演算子フィールド
- 3 1 実行ステップフィールド
- 3 2 入力データ出力データレコード
- 3 3 演算子フィールド
- 3 4 入力データ記憶データレコード
- 3 5 記憶素子フィールド
- 3 6 実行ステップフィールド

【書類名】 図面

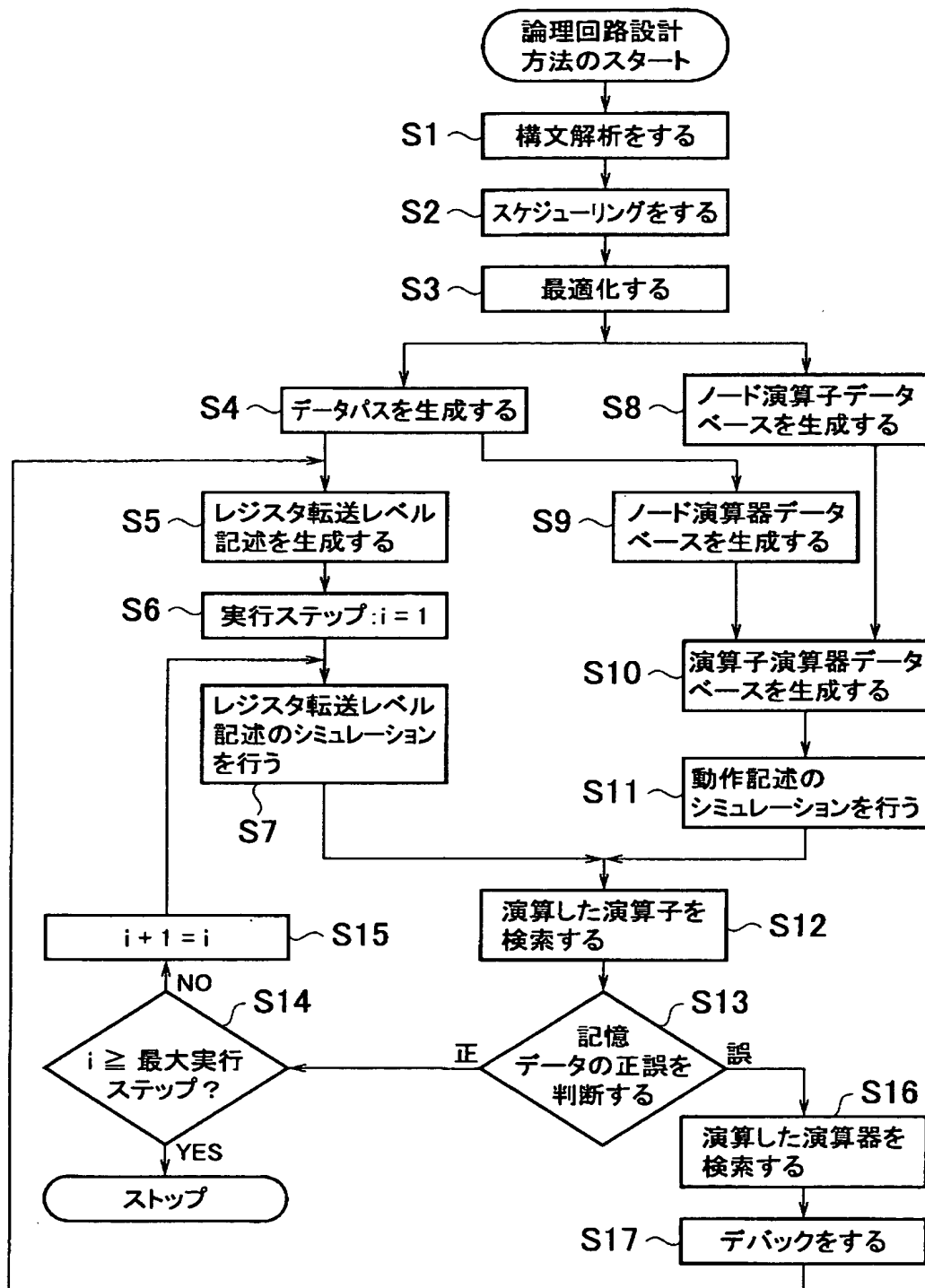
【図 1】



【図 2】



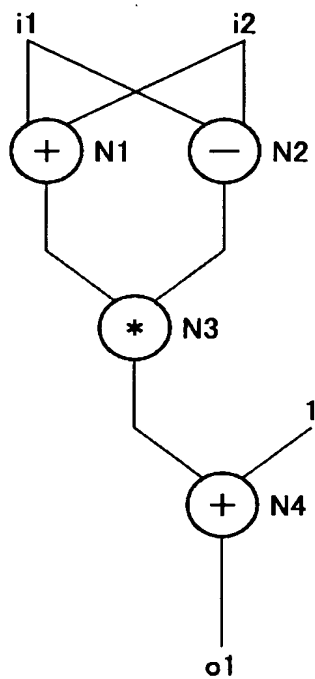
【図 3】



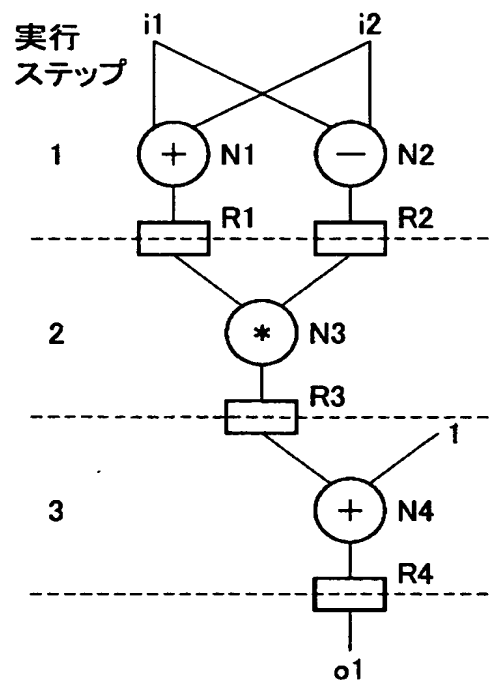
【図 4】

```
void calc(int i1, int i2, int &o1)
{
    o1 = (i1 + i2) * (i1 - i2) + 1;
}
        op1   op2   op3   op4
```

【図 5】



【図 6】



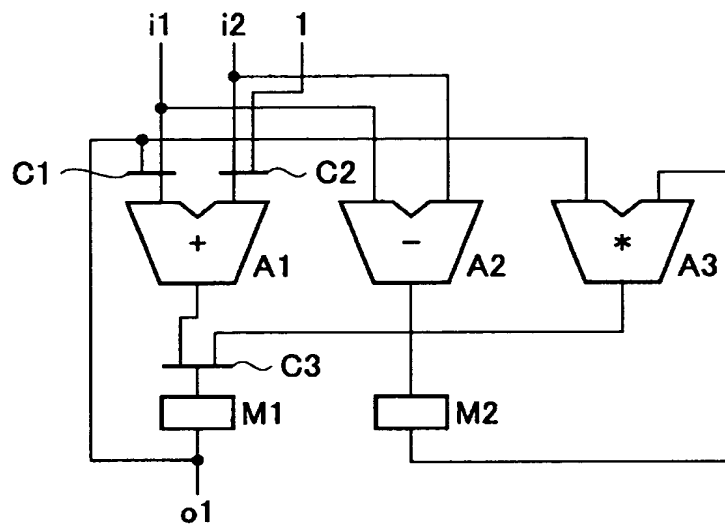
【図 7】

D7

21 {	ノード	演算子
	N1	op1
	N2	op3
	N3	op2
	N4	op4
	R1	op1
	R2	op3
	R3	op2
	R4	op4

22 23

【図 8】



【図 9】

<div style="display: flex; align-items: center;"> 24 { <div style="text-align: center;"> 実行ステップ 演算器 </div> </div>		1	2	3	D8	27
A1		N1		N4		
A2		N2				
A3			N3			
M1		R1	R3	R4		
M2		R2				
25		26				

【図 1 0】

演算器	実行ステップ			D9
	1	2	3	
A1	op1		op4	31
A2	op3			
A3		op2		
M1	op1	op2	op4	
M2	op3			
29		30		

【図 1 1】

(i1, i2)	演算子				33
	op1	op2	op3	op4	
(2, 1)	3	3	1	4	32
(3, 2)	5	5	1	6	
(2, 3)	5	-5	-1	-4	
D13		D10			

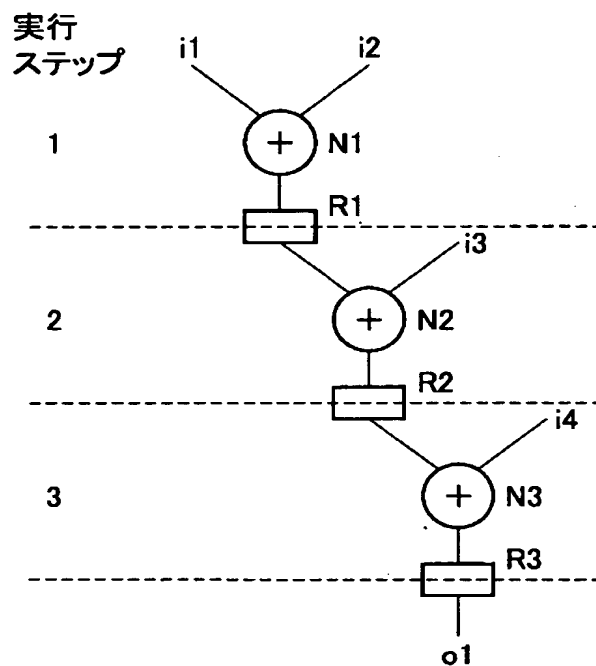
【図 1 2】

(i1, i2)	実行ステップ				36
		1	2	3	
(2, 1)	M1	3	3	4	34
	M2	1			
(3, 2)	M1	5	5	6	
	M2	1			
(2, 3)	M1	5	1275	1276	
	M2	-1			
D13		35	D11		

【図 1 3】

```
void calc(int i1, int i2, int i3, int i4, int &o1)
{
    o1 = i1 + i2 + i3 + i4;
    op1 op2 op3
}
```

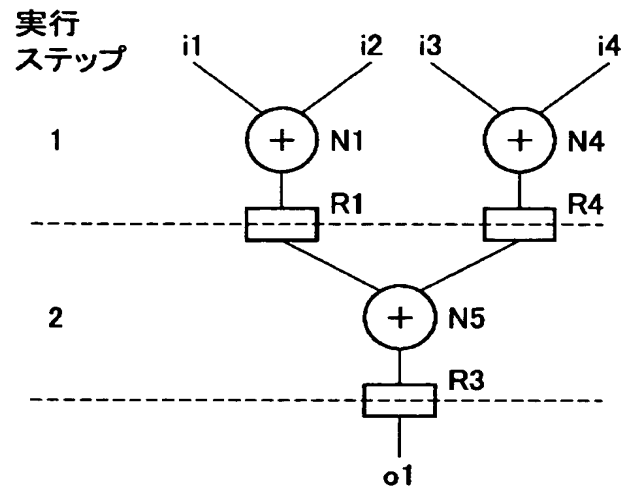
【図 1 4】



【図 1 5】

21 {	ノード	演算子	D7
	N1	op1	
	N2	op2	
	N3	op3	
	R1	op1	
	R2	op2	
	R3	op3	
22		23	

【図 1 6】

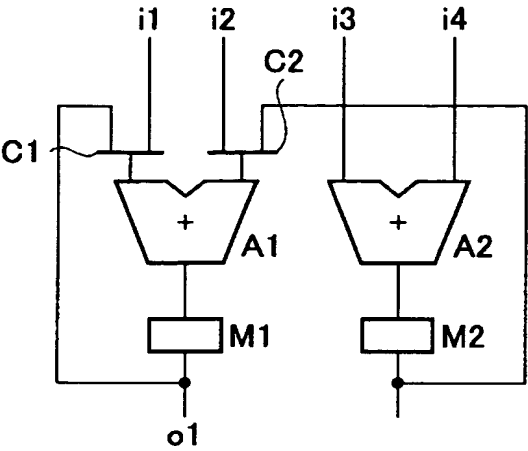


【図 1 7】

ノード	演算子	D7
N1	op1	
N2	op2	
N3	op3	
N4	op2, op3	
N5	op2, op3	
R1	op1	
R2	op2	
R3	op2, op3	
R4	op2, op3	

22 23

【図 1 8】



【図 1 9】

演算器	実行ステップ	
	1	2
A1	N1	N5
A2	N4	
M1	R1	R3
M2	R4	

24 { 25 26 D8 27

【図 2 0】

演算器	実行ステップ	
	1	2
A1	op1	op2, op3
A2	op2, op3	
M1	op1	op2, op3
M2	op2, op3	

28 { 29 30 D9 31

【書類名】 要約書

【要約】

【課題】 動作記述から変換された R T L 記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計方法を提供する。

【解決手段】 論理回路のアルゴリズムを、複数の演算子を有する動作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換し、データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入する。演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成する。演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成する。

【選択図】 図 3



出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 3 0 7 8]

1. 変更年月日 2 0 0 1 年 7 月 2 日
[変更理由] 住所変更
住 所 東京都港区芝浦一丁目 1 番 1 号
氏 名 株式会社東芝